DATA 621 Homework #3

Code

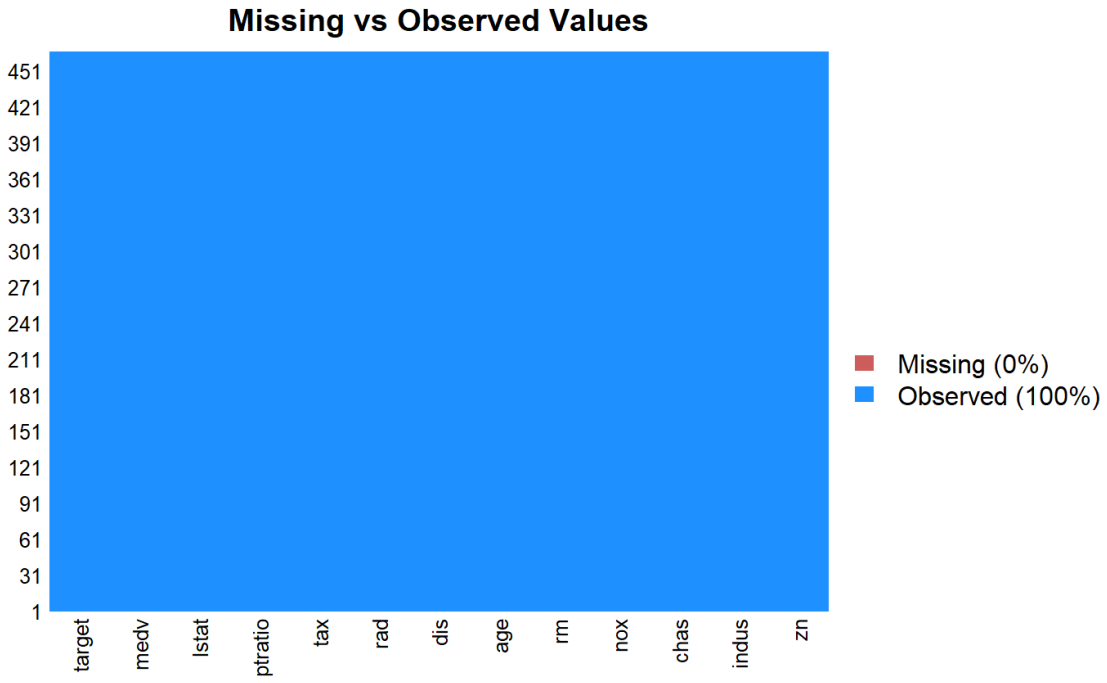
Code

Data Exploration

Are There Missing Values?

First we look at the data to see if any variables have missing data:

Code



It looks like we have a complete data set. No need to impute values.

Splitting the Data

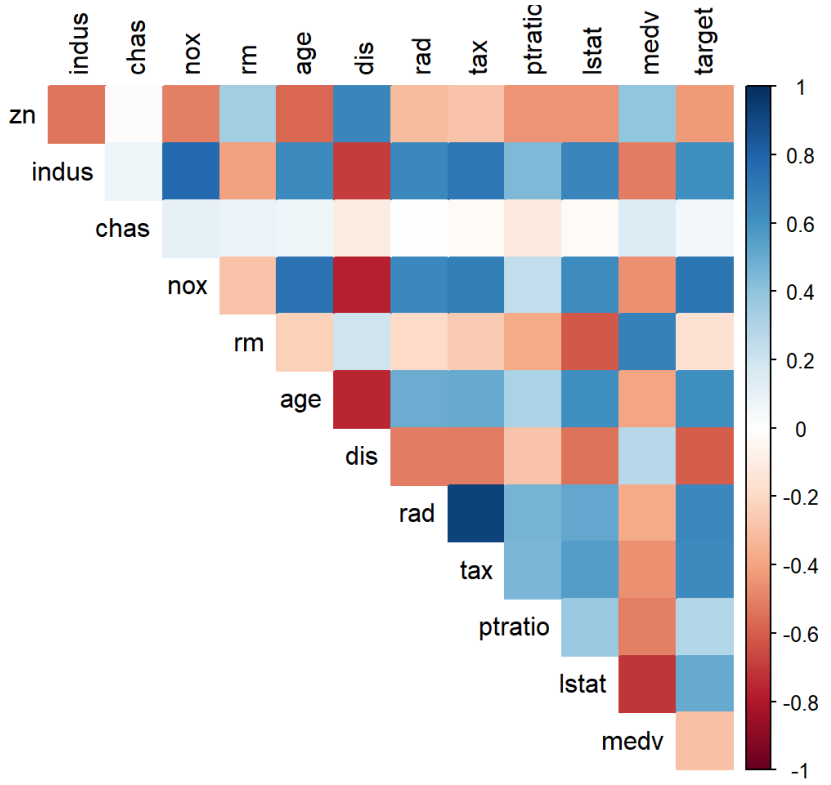
Next, we look to split our data between a training set (`train`) and a test data set (`test`). We'll use a 70-30 split between train and test, respectively.

Code

Exploratory Data Analysis

Now, let's look at our training data. By looking at a correlation matrix, we can see which variables may be too correlated to be included together in a model as predictor variables. This will help us later during the model selection process.

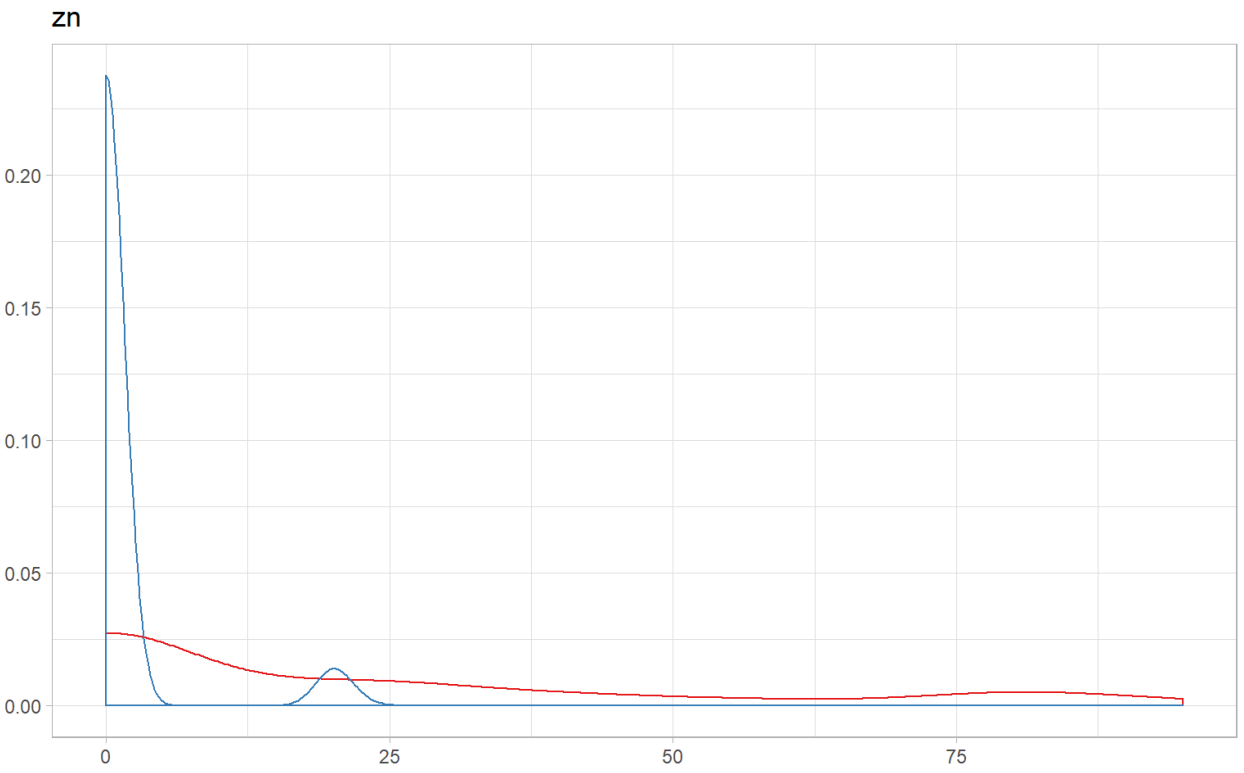
Code

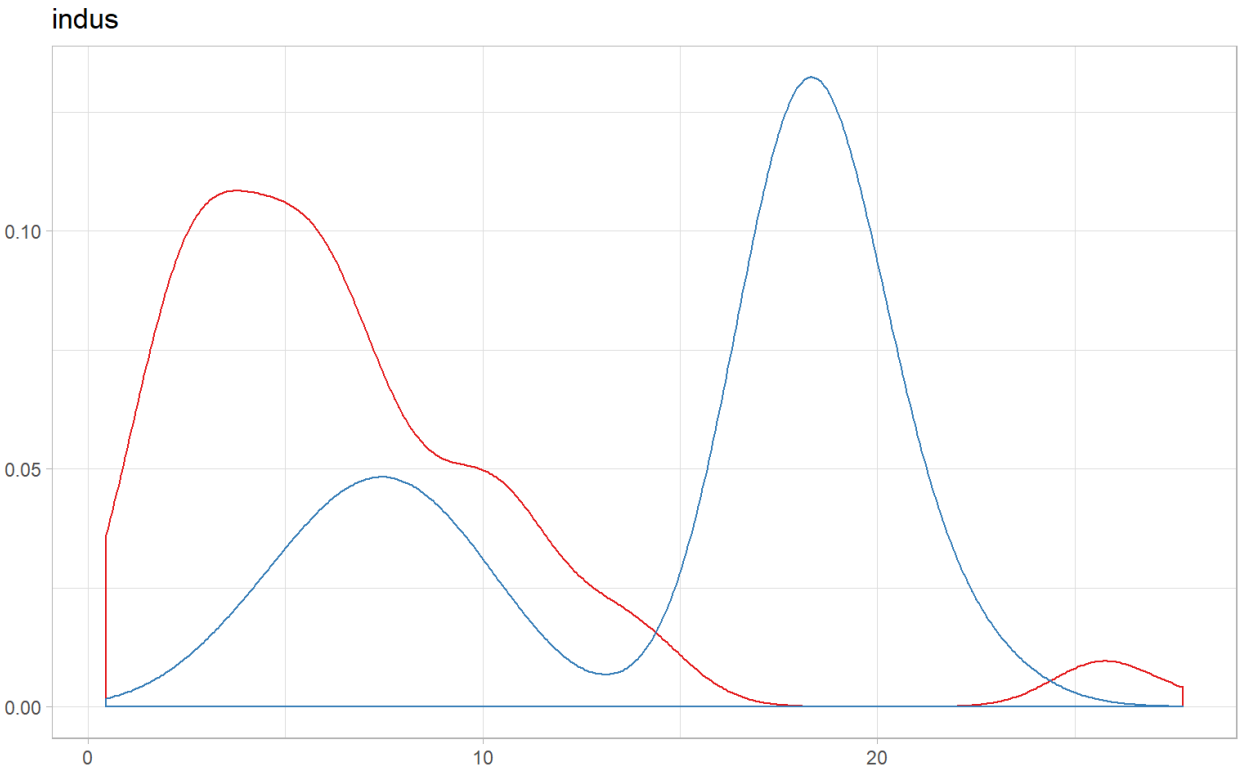


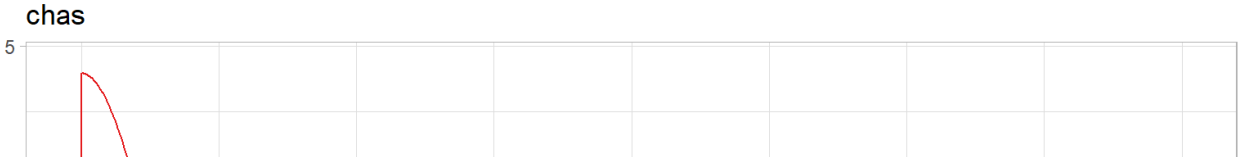
Next we will look at each potential predictor and how it is distributed across the target variable.

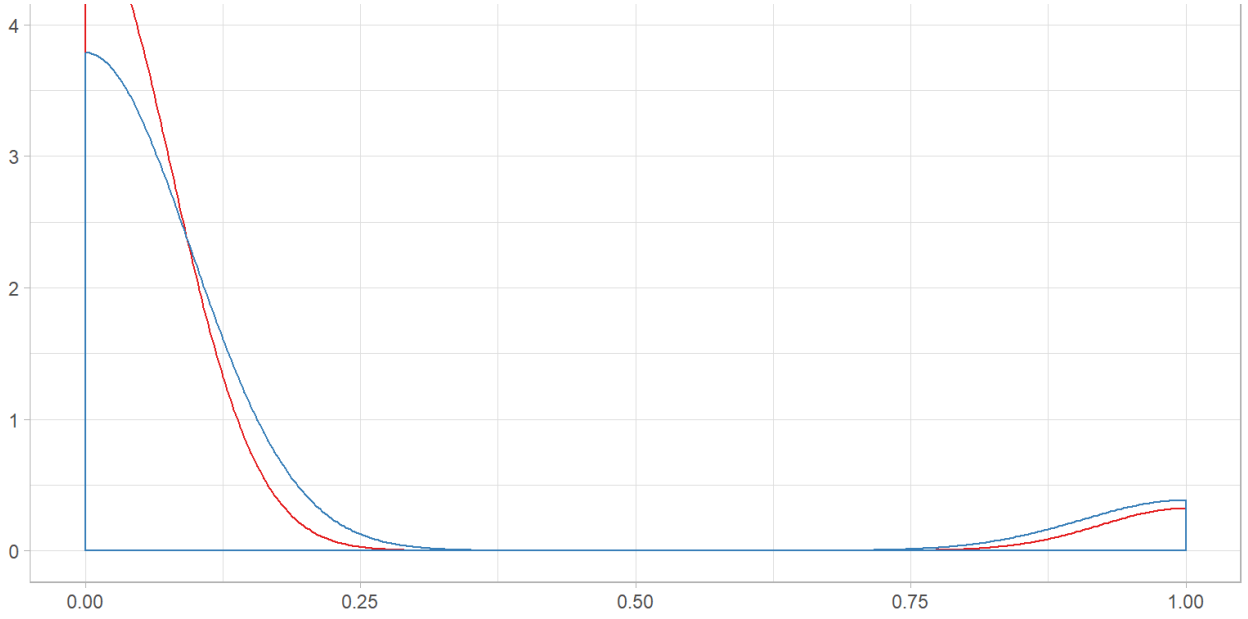
The following plots show how predictors are distributed between a positive target variable (areas with crime rates higher than the median, i.e. blue) and a negative target variable (areas with crime rates below the median, i.e. red). What we are looking for is variables that show way to split data into two groups.

Code

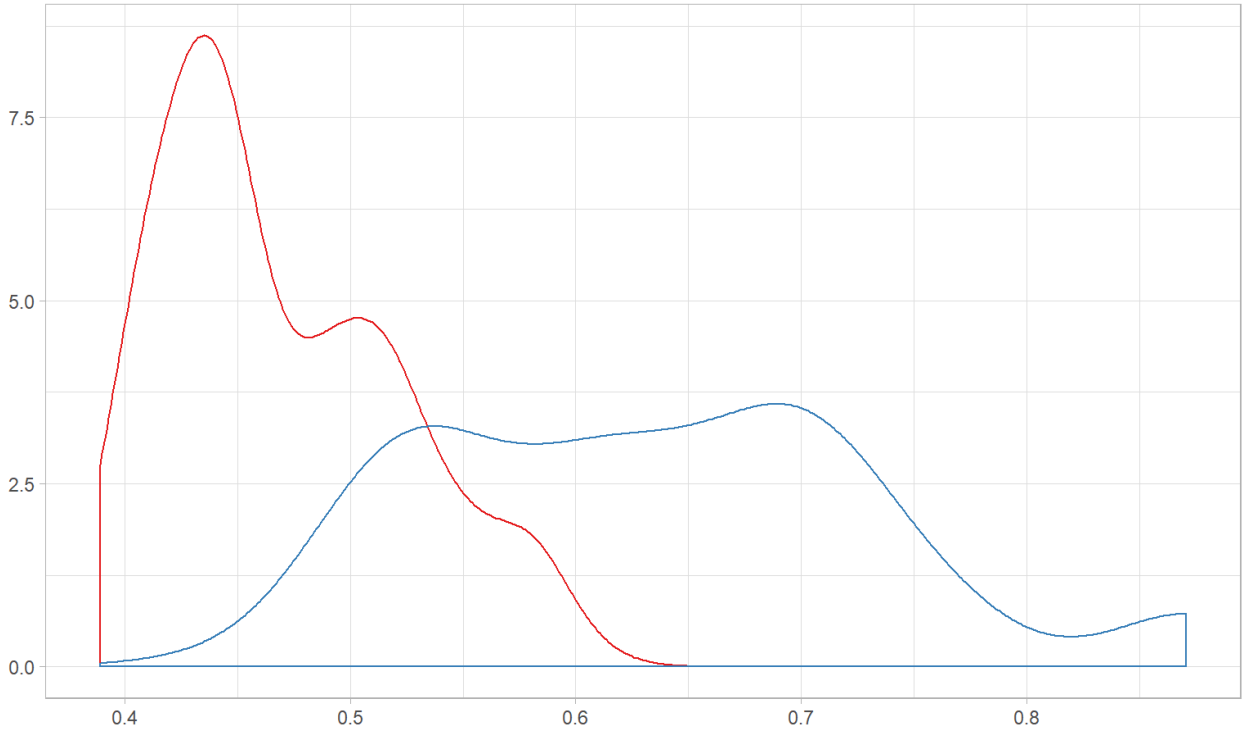

zn
0.20
0.15
0.10
0.05
0.00
0
25
50
75

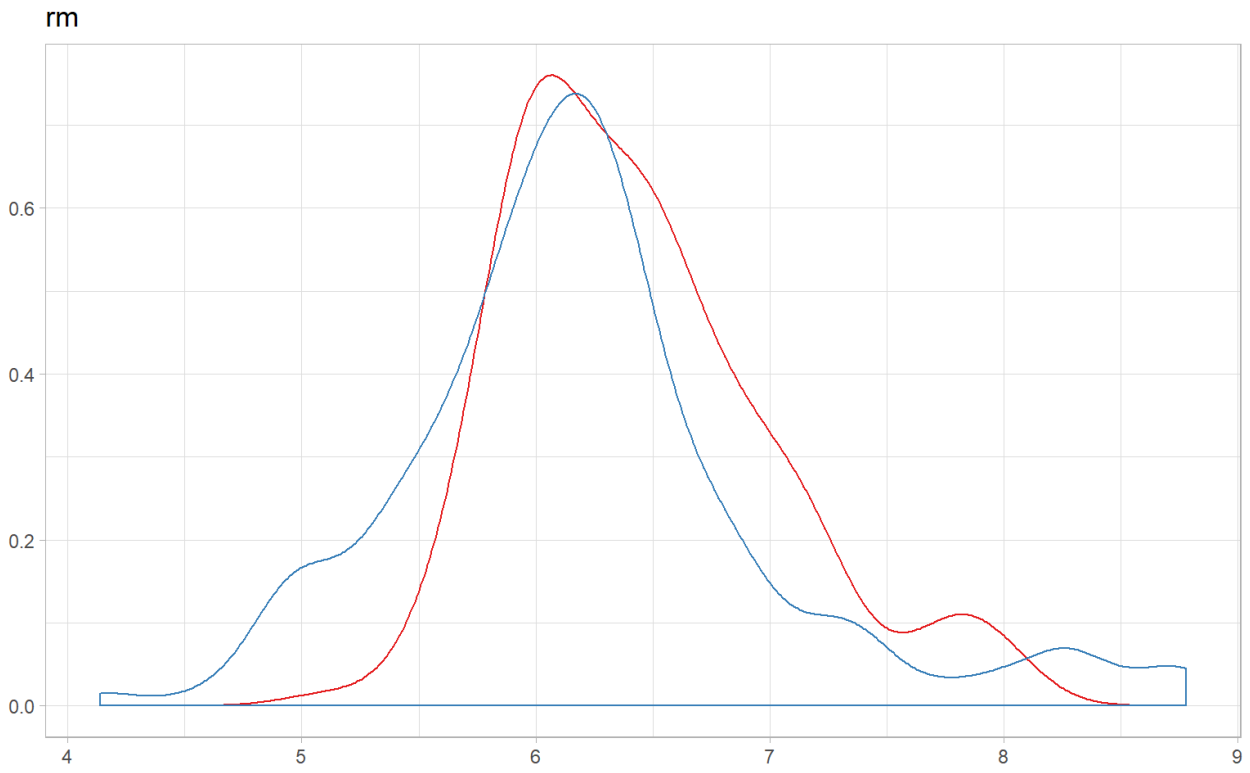

indus
0.10
0.05
0.00
0
10
20

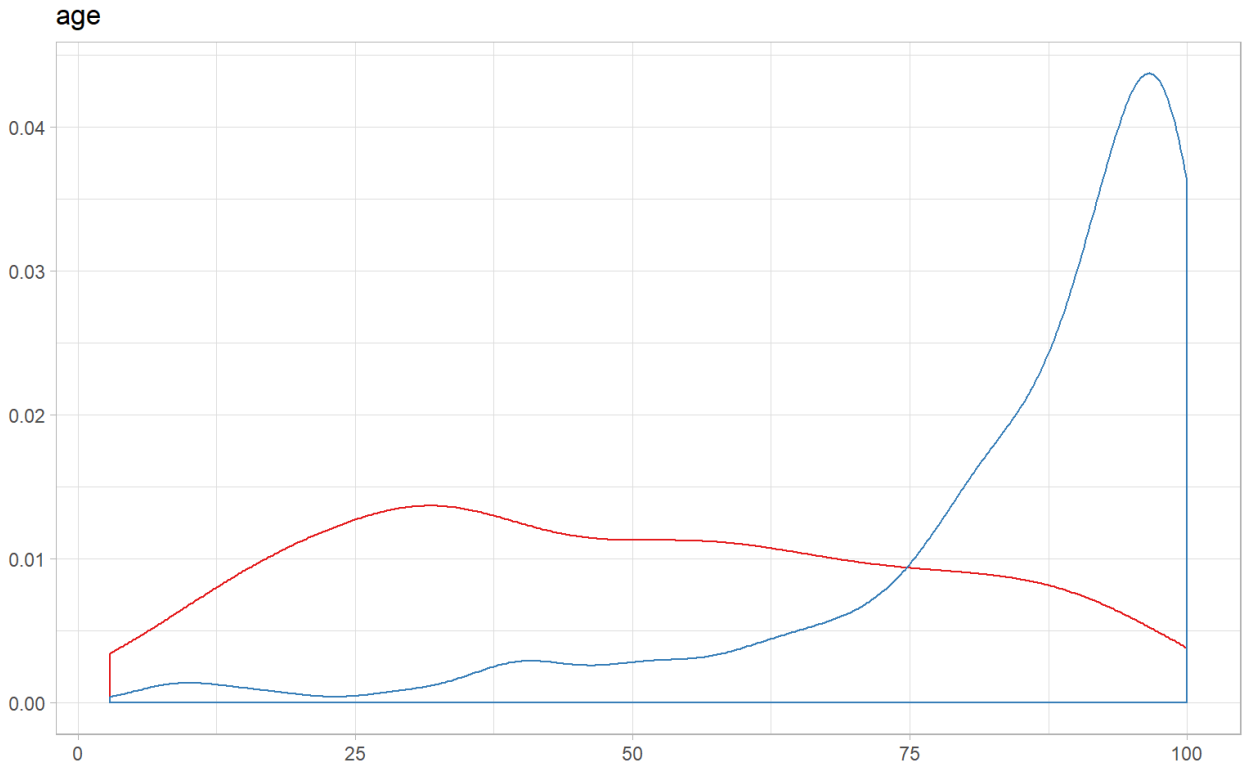

chas
5

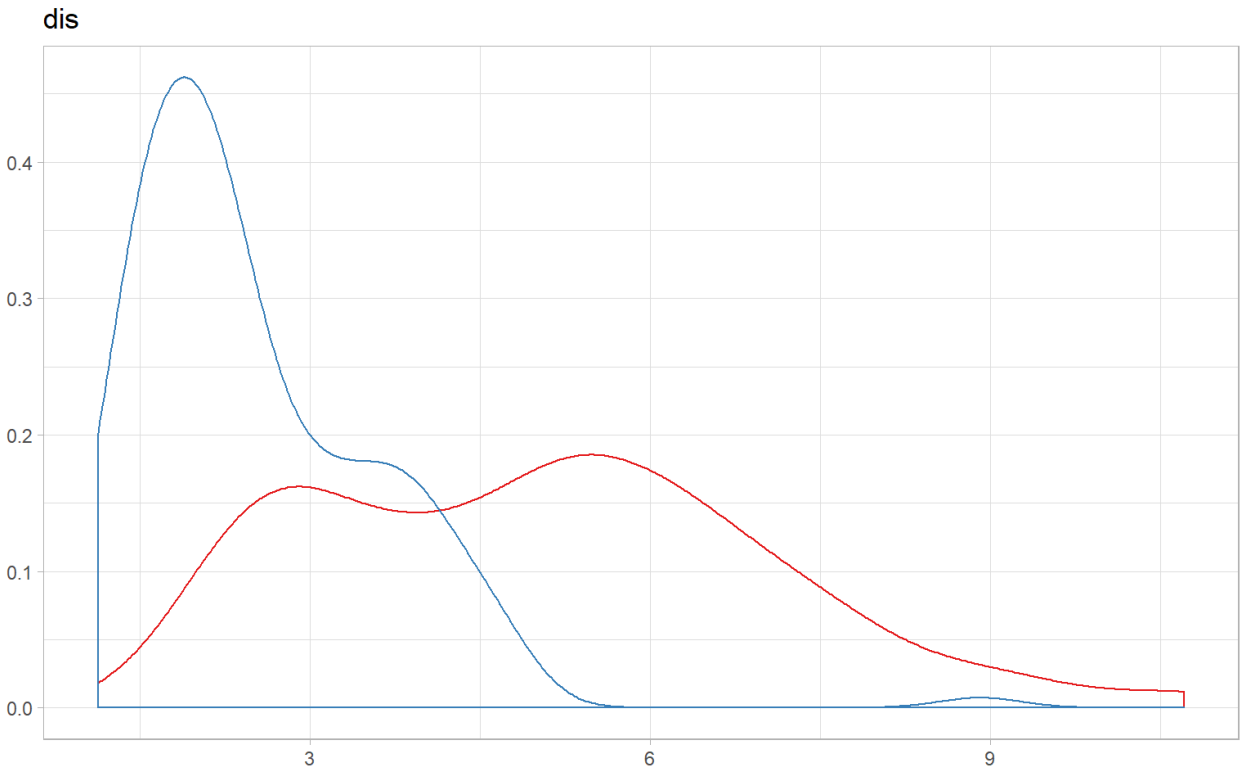


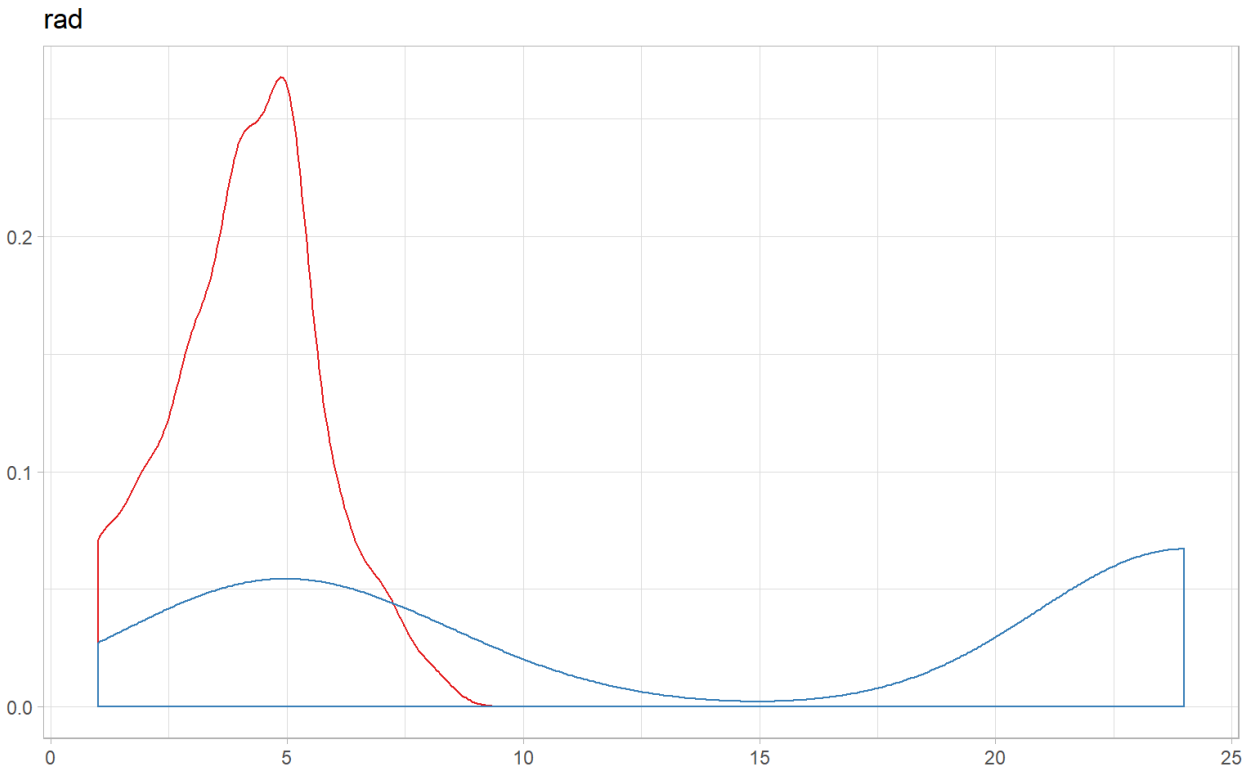
nox

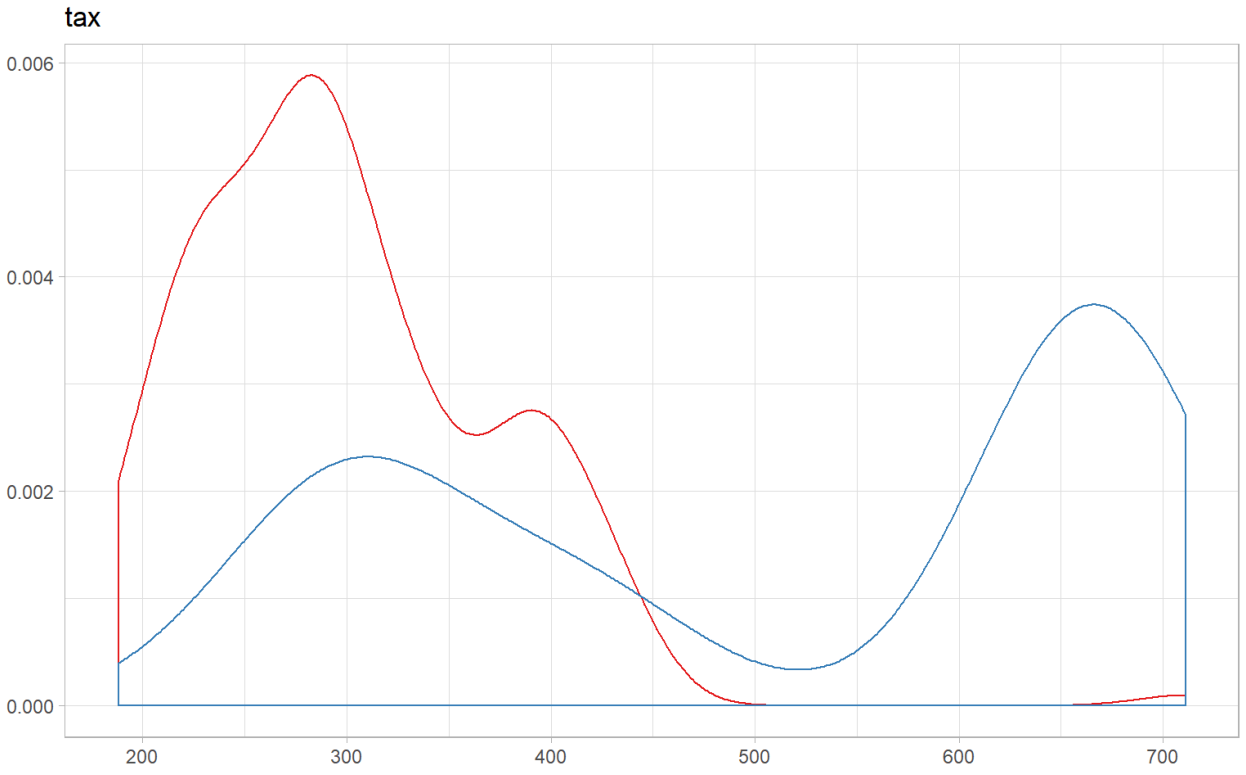


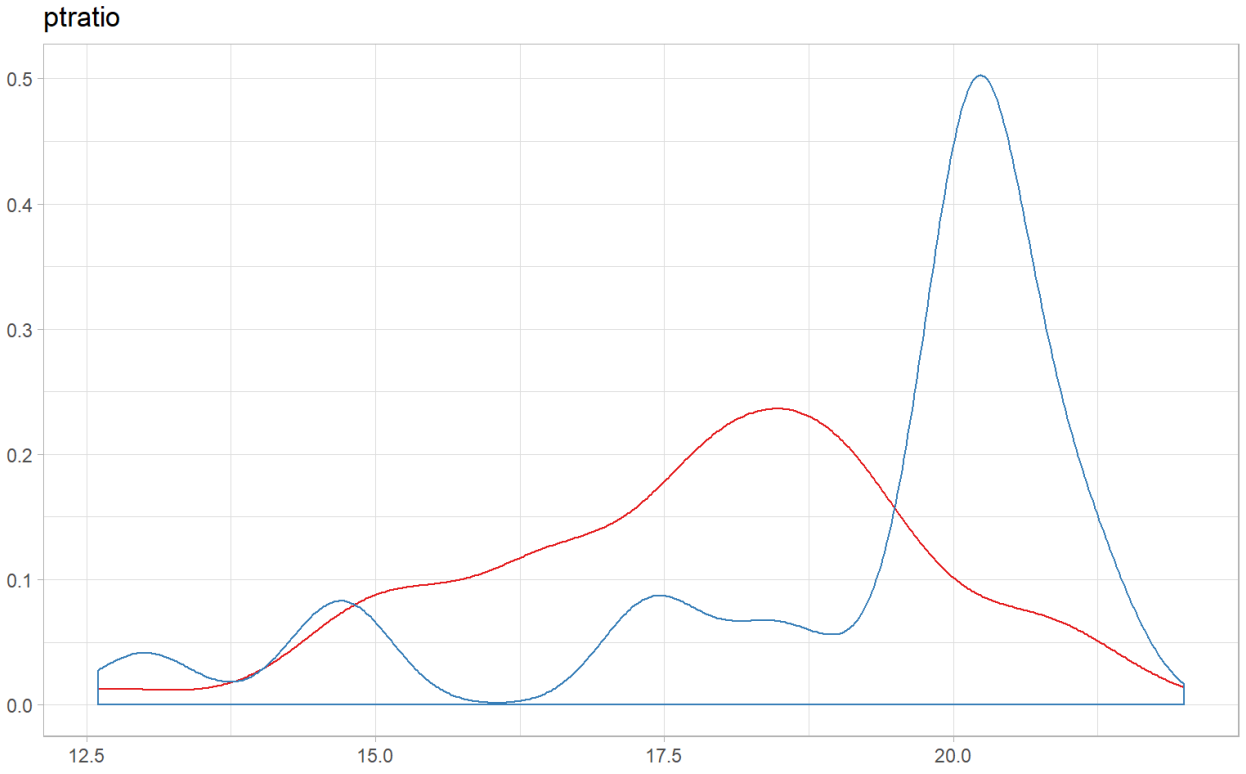

rm
0.6
0.4
0.2
0.0
4
5
6
7
8
9

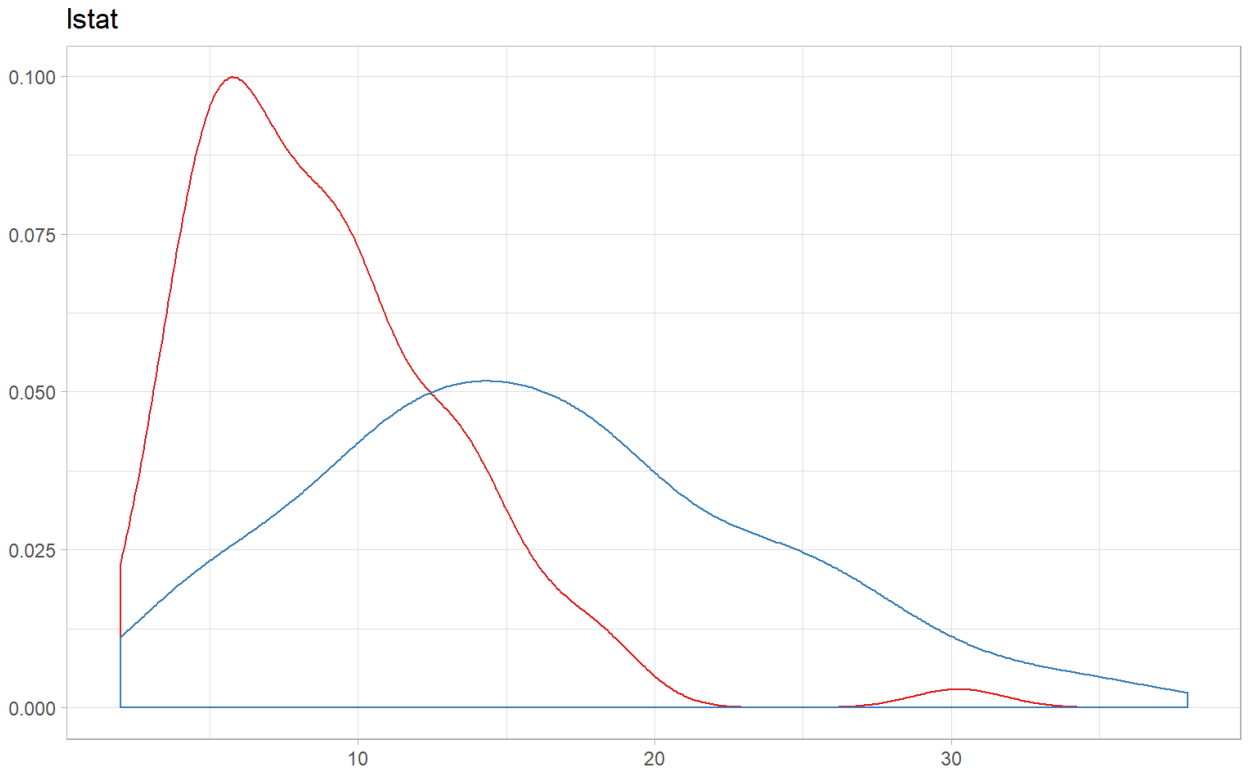

age
0.04
0.03
0.02
0.01
0.00
0
25
50
75
100

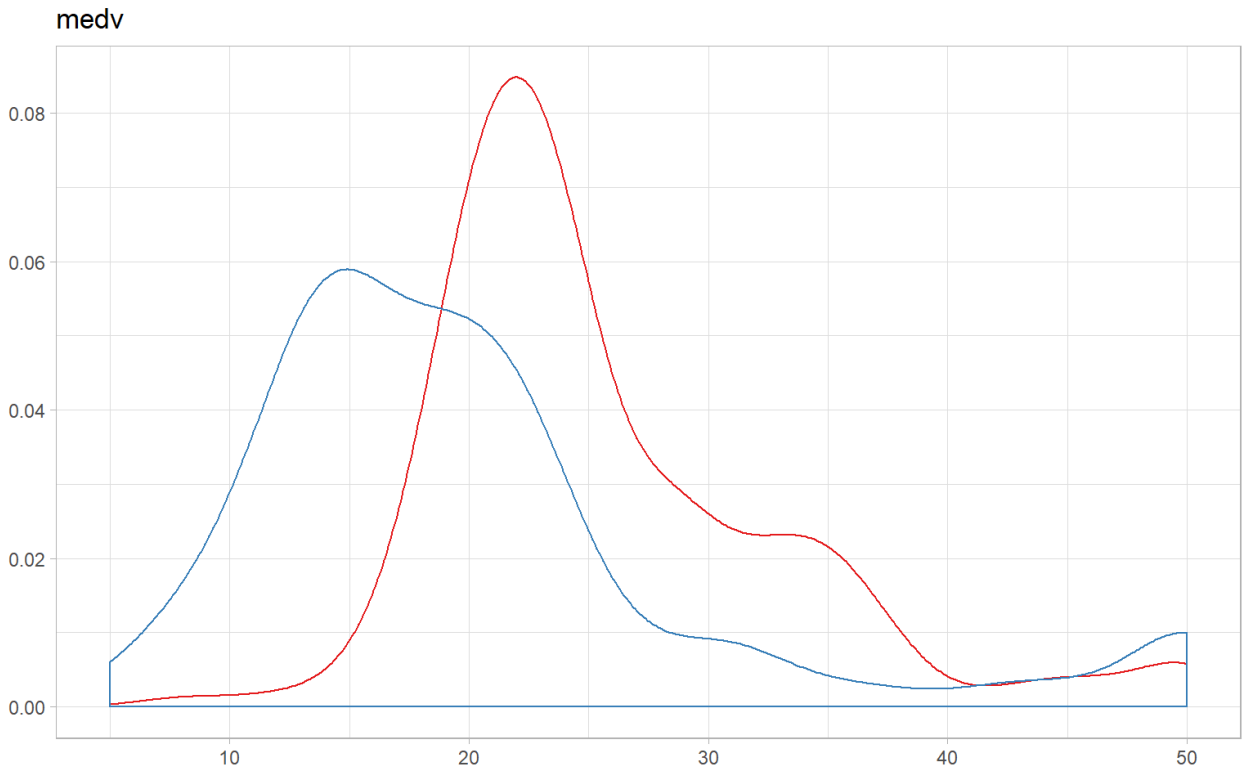

dis
0.4
0.3
0.2
0.1
0.0
3
6
9


rad
0.2
0.1
0.0
0
5
10
15
20
25


tax
0.006
0.004
0.002
0.000
200
300
400
500
600
700


ptratio
0.5
0.4
0.3
0.2
0.1
0.0
12.5
15.0
17.5
20.0





Looking at the plots above, the `nox` variable seems to be the best variable to divide the data into the two groups.

Basic Model Building

We start by applying occam's razor and create a baseline model that only has one predictor. Any model we build beyond that will have to outperform this simplest model.

Code

```
Call:
glm(formula = target ~ nox, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1163  -0.4050  -0.1814   0.2717   2.5754

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -16.370      1.823  -8.981   <2e-16 ***
nox           30.372      3.444   8.818   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 451.70  on 326  degrees of freedom
Residual deviance: 205.67  on 325  degrees of freedom
AIC: 209.67

Number of Fisher Scoring iterations: 6
```

Code

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 50 11
         1 12 66

               Accuracy : 0.8345
                 95% CI : (0.7621, 0.8921)
    No Information Rate : 0.554
    P-Value [Acc > NIR] : 2.178e-12

                  Kappa : 0.6646

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8571
            Specificity : 0.8065
         Pos Pred Value : 0.8462
         Neg Pred Value : 0.8197
             Prevalence : 0.5540
         Detection Rate : 0.4748
   Detection Prevalence : 0.5612
      Balanced Accuracy : 0.8318

       'Positive' Class : 1
```

Our baseline model is ok with an F1 score of 0.8516129.

Next we try adding every other variable, to build a full model. From here we can work backwards and eliminate non-significant predictors:

Code

```
Call:
glm(formula = target ~ ., family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1095  -0.1955  -0.0025   0.0010   3.4349

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -45.183151   8.234576  -5.487 4.09e-08 ***
zn           -0.075536   0.044071  -1.714 0.086539 .
indus        -0.091581   0.057306  -1.598 0.110020
chas          0.686033   0.878327   0.781 0.434763
nox          52.841701   9.811850   5.385 7.22e-08 ***
rm           -0.060835   0.895649  -0.068 0.945847
age           0.014775   0.014866   0.994 0.320276
dis           0.748904   0.277029   2.703 0.006865 **
rad           0.693492   0.192573   3.601 0.000317 ***
tax          -0.004634   0.003734  -1.241 0.214522
ptratio       0.376793   0.162243   2.322 0.020211 *
lstat         0.146590   0.065105   2.252 0.024349 *
medv          0.159840   0.086270   1.853 0.063912 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 451.70  on 326  degrees of freedom
Residual deviance: 136.79  on 314  degrees of freedom
AIC: 162.79

Number of Fisher Scoring iterations: 9
```

Code

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 55  6
         1  7 71

               Accuracy : 0.9065
                 95% CI : (0.8454, 0.9493)
    No Information Rate : 0.554
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8104

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9221
            Specificity : 0.8871
         Pos Pred Value : 0.9103
         Neg Pred Value : 0.9016
             Prevalence : 0.5540
         Detection Rate : 0.5108
   Detection Prevalence : 0.5612
      Balanced Accuracy : 0.9046

       'Positive' Class : 1
```

The full model has an F1 score is 0.916129, which is a bit higher than before. However, many variables do not seem to be significant.

After some backward elimination of non-significant predictor variables, we arrive at the following model:

Code

```
Call:
glm(formula = target ~ . - tax - rm - chas - age - zn - indus,
    family = binomial(link = "logit"), data = train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.25157  -0.28019  -0.06287   0.00092   2.93960

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -40.36177    7.32297  -5.512 3.55e-08 ***
nox          43.62710    7.91385   5.513 3.53e-08 ***
dis           0.43922    0.19736   2.225  0.02605 *
rad           0.64560    0.14500   4.453 8.49e-06 ***
ptratio       0.38903    0.13439   2.895  0.00379 **
lstat         0.13126    0.05636   2.329  0.01986 *
medv          0.14156    0.04686   3.021  0.00252 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 451.70  on 326  degrees of freedom
Residual deviance: 149.16  on 320  degrees of freedom
AIC: 163.16

Number of Fisher Scoring iterations: 9
```

Code

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 51  5
         1 11 72

               Accuracy : 0.8849
                 95% CI : (0.8198, 0.9328)
    No Information Rate : 0.554
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7649

 Mcnemar's Test P-Value : 0.2113

            Sensitivity : 0.9351
            Specificity : 0.8226
         Pos Pred Value : 0.8675
         Neg Pred Value : 0.9107
             Prevalence : 0.5540
         Detection Rate : 0.5180
   Detection Prevalence : 0.5971
      Balanced Accuracy : 0.8788

       'Positive' Class : 1
```

Alternative Models

Feature Engineering

In this portion below we explored using the difference between the class distributions as a feature itself. So instead of the raw feature itself the we created density features that are essientially the differences in probability of the positve and negative classes at that certain point for the feature.

Code

Creating Some New Variables

Code

```
'data.frame':   327 obs. of  13 variables:
 $ zn     : num  0 0 0 30 0 0 80 22 0 22 ...
 $ indus  : num  19.58 19.58 18.1 4.93 2.46 ...
 $ chas   : int  0 1 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.605 0.871 0.74 0.428 0.488 0.52 0.392 0.431 0.437 0.431 ...
 $ rm     : num  7.93 5.4 6.49 6.39 7.16 ...
 $ age    : num  96.2 100 100 7.8 92.2 71.3 19.1 8.9 45 8.4 ...
 $ dis    : num  2.05 1.32 1.98 7.04 2.7 ...
 $ rad    : int  5 5 24 6 3 5 1 7 5 7 ...
 $ tax    : int  403 403 666 300 193 384 315 330 398 330 ...
 $ ptratio: num  14.7 14.7 20.2 16.6 17.8 20.9 16.4 19.1 18.7 19.1 ...
 $ lstat  : num  3.7 26.82 18.85 5.19 4.82 ...
 $ medv   : num  50 13.4 15.4 23.7 37.9 26.5 20.9 24.8 21.4 42.8 ...
 $ target : int  1 1 1 0 0 0 0 0 0 1 ...
```

Code

```
'data.frame':   139 obs. of  16 variables:
 $ zn       : num  0 0 0 0 0 100 0 0 0 0 ...
 $ indus    : num  18.1 18.1 5.19 18.1 2.46 1.32 18.1 3.24 6.2 2.89 ...
 $ chas     : int  0 0 0 0 0 0 0 0 1 0 ...
 $ nox      : num  0.693 0.693 0.515 0.532 0.488 0.411 0.679 0.46 0.507 0.445 ...
 $ rm       : num  5.45 4.52 6.32 7.06 6.15 ...
 $ age      : num  100 100 38.1 77 68.8 40.5 95.4 32.2 66.5 62.5 ...
 $ dis      : num  1.49 1.66 6.46 3.41 3.28 ...
 $ rad      : int  24 24 5 24 3 5 24 4 8 2 ...
 $ tax      : int  666 666 224 666 193 256 666 430 307 276 ...
 $ ptratio  : num  20.2 20.2 20.2 20.2 17.8 15.1 20.2 16.9 17.4 18 ...
 $ lstat    : num  30.59 36.98 5.68 7.01 13.15 ...
 $ medv     : num  5 7 22.2 25 29.6 31.6 8.3 19.8 29 33.2 ...
 $ target   : int  1 1 0 1 0 0 1 0 1 0 ...
 $ baseline : num  1 1 0 0 0 0 1 0 0 0 ...
 $ fullmodel: num  1 1 1 1 0 0 1 0 1 0 ...
 $ model1   : num  1 1 0 1 0 0 1 0 1 0 ...
```

Code

```
'data.frame':   327 obs. of  22 variables:
 $ zn             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ chas           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ rm             : num  5.68 5.99 5.28 5 6.78 ...
 $ target         : int  1 1 1 1 1 0 1 1 1 1 ...
 $ nox_density    : num  1.73 1.73 1.7 1.7 1.71 ...
 $ nox            : num  0.693 0.693 0.7 0.7 0.679 0.609 0.718 0.693 0.74 0.679 ...
 $ age_density    : num  3.15 3.15 3.65 2.04 2.41 ...
 $ age            : num  100 100 98.1 89.5 90.8 98 76.5 85.4 100 95.6 ...
 $ indus_density  : num  3.58 3.58 3.58 3.58 3.58 ...
 $ indus          : num  18.1 18.1 18.1 18.1 18.1 ...
 $ dis_density    : num  3.02 3.47 3.02 3.31 3.64 ...
 $ dis            : num  1.43 1.59 1.43 1.52 1.82 ...
 $ rad_density    : num  1.54 1.54 1.54 1.54 1.54 ...
 $ rad            : int  24 24 24 24 24 4 24 24 24 24 ...
 $ tax_density    : num  1.95 1.95 1.95 1.95 1.95 ...
 $ tax            : int  666 666 666 666 666 711 666 666 666 666 ...
 $ ptratio_density: num  3.87 3.87 3.87 3.87 3.87 ...
 $ ptratio        : num  20.2 20.2 20.2 20.2 20.2 20.1 20.2 20.2 20.2 20.2 ...
 $ lstat_density  : num  1.013 0.723 0.248 0.222 0.822 ...
 $ lstat          : num  23 26.8 30.8 32 25.8 ...
 $ medv_density   : num  0.258 0.325 0.546 0.576 0.591 ...
 $ medv           : num  5 5.6 7.2 7.4 7.5 8.1 8.4 8.5 8.7 9.5 ...
```

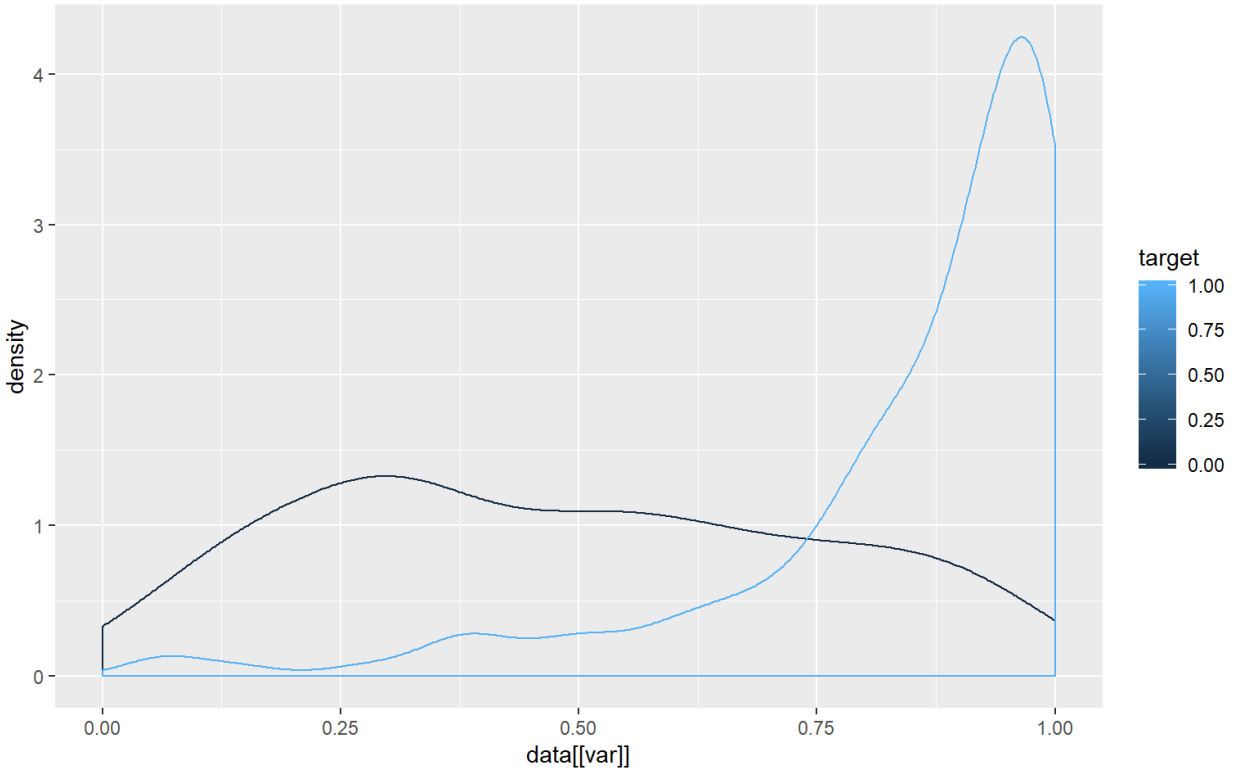
Code

```
'data.frame':   139 obs. of  25 variables:
 $ zn             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ chas           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ rm             : num  5.45 5.85 5.41 4.52 6.43 ...
 $ target         : int  1 1 0 1 1 1 1 1 1 1 ...
 $ baseline       : num  1 1 1 1 1 1 1 1 1 1 ...
 $ fullmodel      : num  1 1 0 1 1 1 1 1 1 1 ...
 $ model1         : num  1 1 1 1 1 1 1 1 1 1 ...
 $ nox_density    : num  1.73 1.73 1.2 1.73 1.7 ...
 $ nox            : num  0.693 0.693 0.609 0.693 0.679 0.693 0.679 0.693 0.693 0.7 ...
 $ age_density    : num  3.15 0.24 3.62 3.15 3.15 ...
 $ age            : num  100 77.8 98.3 100 100 100 95.4 96 98.9 91.2 ...
 $ indus_density  : num  3.47 3.47 -0.11 3.47 3.47 ...
 $ indus          : num  18.1 18.1 27.7 18.1 18.1 ...
 $ dis_density    : num  2.96 2.96 3.55 3.36 3.63 ...
 $ dis            : num  1.49 1.5 1.76 1.66 1.83 ...
 $ rad_density    : num  1.54 1.54 -4.35 1.54 1.54 ...
 $ rad            : int  24 24 4 24 24 24 24 24 24 24 ...
 $ tax_density    : num  1.95 1.95 1.37 1.95 1.95 ...
 $ tax            : int  666 666 711 666 666 666 666 666 666 666 ...
 $ ptratio_density: num  3.73 3.73 3.31 3.73 3.73 ...
 $ ptratio        : num  20.2 20.2 20.1 20.2 20.2 20.2 20.2 20.2 20.2 20.2 ...
 $ lstat_density  : num  0.2284 0.2481 0.9091 0.0852 0.3155 ...
 $ lstat          : num  30.6 30 24 37 29.1 ...
 $ medv_density   : num  0.258 0.414 0.517 0.517 0.546 ...
 $ medv           : num  5 6.3 7 7 7.2 7.2 8.3 8.3 8.5 8.8 ...
```

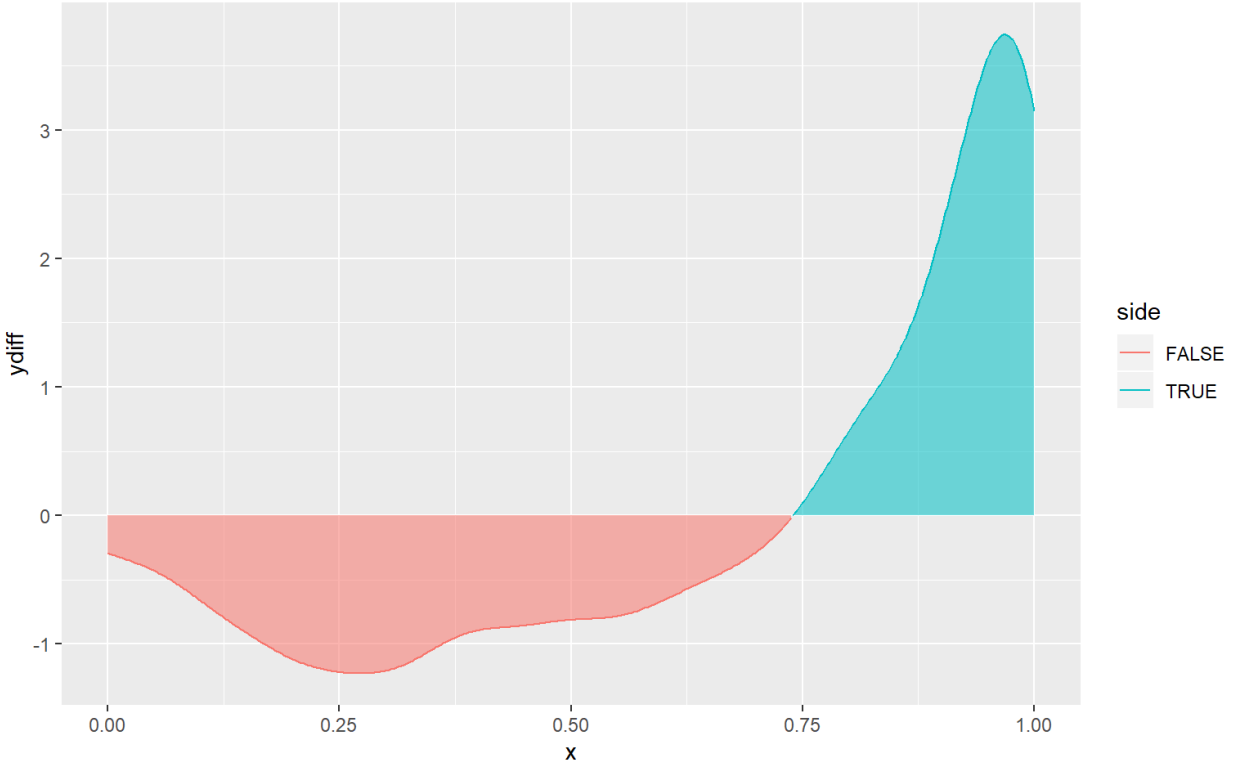
Example of the new meta feature

The first graph below is the difference between the distributions and the second graph is the new derived predictive feature

Code


density
data[[var]]
target
1.00
0.75
0.50
0.25
0.00

Code


ydiff
x
side
FALSE
TRUE

With our new density variables, we can run another model:

Code

```
Call:
glm(formula = target ~ nox_density + indus_density + age_density +
    dis_density + rad_density + tax_density + ptratio_density +
    lstat_density + medv_density, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.96836  -0.31245  -0.06225   0.20194   3.01402

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)       0.8093     0.5320   1.521 0.128210
nox_density       1.1492     0.2600   4.420 9.86e-06 ***
indus_density     1.0706     0.2488   4.302 1.69e-05 ***
age_density       0.1219     0.1698   0.718 0.472916
dis_density      -0.1849     0.2138  -0.865 0.387284
rad_density       0.4141     0.1226   3.379 0.000727 ***
tax_density      -1.2582     0.4408  -2.854 0.004314 **
ptratio_density   0.1550     0.1524   1.017 0.309139
lstat_density    -0.2456     0.2055  -1.195 0.231963
medv_density      0.5417     0.2062   2.628 0.008596 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 451.70  on 326  degrees of freedom
Residual deviance: 164.12  on 317  degrees of freedom
AIC: 184.12

Number of Fisher Scoring iterations: 7
```

Code

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 57 11
         1  5 66

               Accuracy : 0.8849
                 95% CI : (0.8198, 0.9328)
    No Information Rate : 0.554
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7692

 Mcnemar's Test P-Value : 0.2113

            Sensitivity : 0.9194
            Specificity : 0.8571
         Pos Pred Value : 0.8382
         Neg Pred Value : 0.9296
             Prevalence : 0.4460
         Detection Rate : 0.4101
   Detection Prevalence : 0.4892
      Balanced Accuracy : 0.8882

       'Positive' Class : 0
```

Model Selection

Since the assignment mentions that the purpose is prediction, we will prefer F1 score as our measure of model success.

Code

model	predictors	F1	deviance	r2	aic
baseline	1	0.8516129	205.6730	0.5446683	209.6730
fullmodel	12	0.9161290	136.7893	0.6971673	162.7893
model1	6	0.9000000	149.1603	0.6697795	163.1603
density models	9	0.8769231	164.1193	0.6366623	184.1193

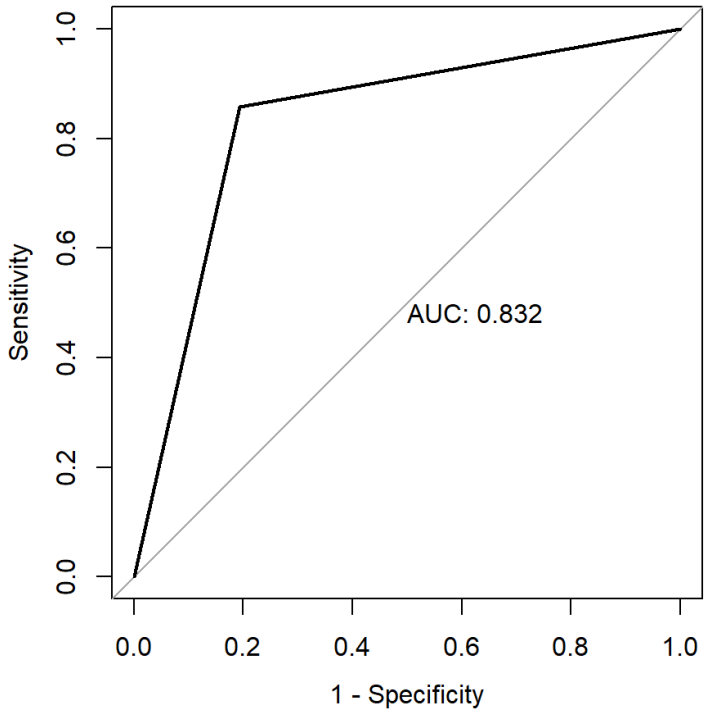
Looking the three models together, `model1` looks like the best one. Although the full model scored an F1 that was ever-so-slightly higher on our test data set, `model1` has half the predictors and it's scores are almost exactly the same as `fullmodel`.

pROC Output

ROC curves can give us another look at which model might be better suited for prediction.

Baseline

Code



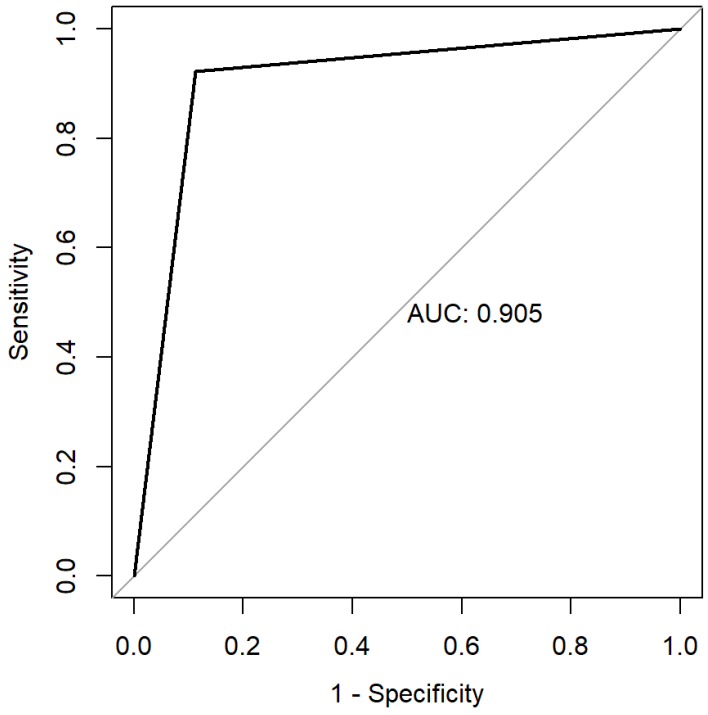

```
Call:
roc.default(response = test[["target"]], predictor = test[["baseline"]],     plot = TRUE, legacy

Data: test[["baseline"]] in 62 controls (test[["target"]] 0) < 77 cases (test[["target"]] 1).
Area under the curve: 0.8318
```

Code

Full Model

Code



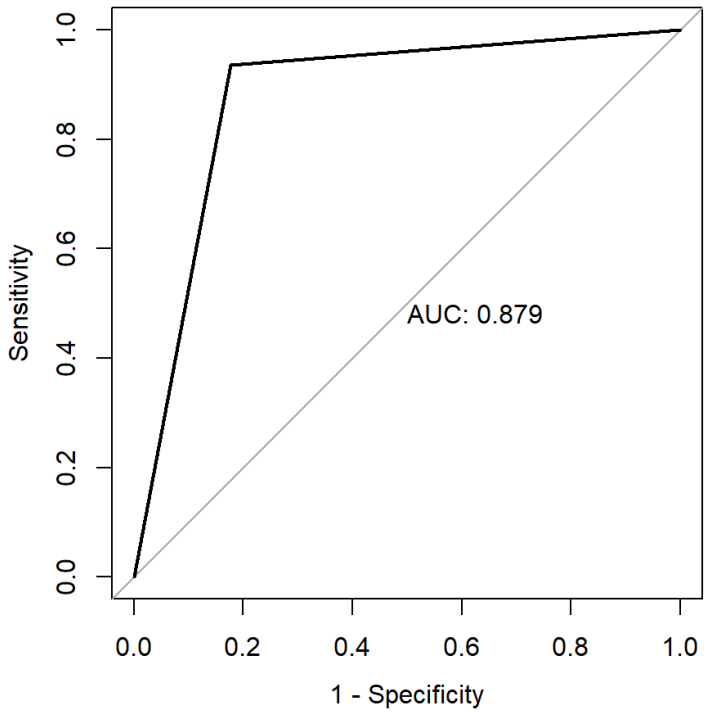

```
Call:
roc.default(response = test[["target"]], predictor = test[["fullmodel"]],     plot = TRUE, legac

Data: test[["fullmodel"]] in 62 controls (test[["target"]] 0) < 77 cases (test[["target"]] 1).
Area under the curve: 0.9046
```

Code

Model1

Code



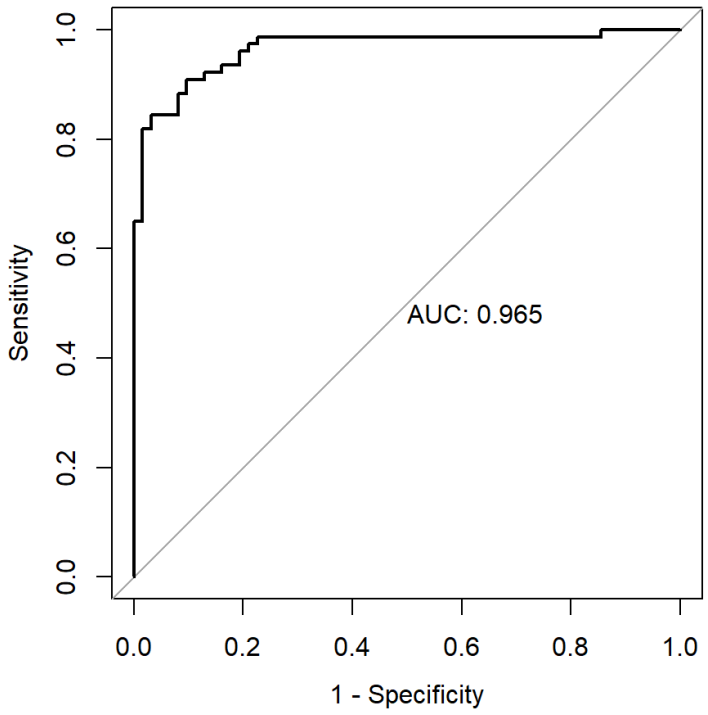

```
Call:
roc.default(response = test[["target"]], predictor = test[["model1"]],     plot = TRUE, legacy.a

Data: test[["model1"]] in 62 controls (test[["target"]] 0) < 77 cases (test[["target"]] 1).
Area under the curve: 0.8788
```

Code

Density Model

Code




```
Call:
roc.default(response = test[["target"]], predictor = test[["density_models_yhat"]],     plot = T

Data: test[["density_models_yhat"]] in 62 controls (test[["target"]] 0) < 77 cases (test[["targe
Area under the curve: 0.9652
```

Code